2016 21st International Conference on Engineering of Complex Computer Systems

Guiding the selection of security patterns for real-time systems

Anas Motii*, Brahim Hamid†, Agnes Lanusse*,Jean-Michel Bruel†
*CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, P.C. 174, Gif-sur-Yvette, 91191, France
†IRIT, University of Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
{anas.motii,agnes.lanusse}@cea.fr, {hamid,bruel}@irit.fr

***Abstract*—Securing critical systems such as Cyber-Physical Systems (CPS) is an important feature especially when it comes to critical transmitted data in a real-time environment. At the same time, the implementation of security counter-measures in such systems may impact transmission delays of critical tasks. For this reason selecting proper security mechanisms in such critical systems is an important issue. In this context, we propose a model-based approach for selecting proper security solution alternatives composed of security patterns at early design stage against real-time requirements. We provide a generalizable and tool-supported solution to support the approach using UML and its profiles. A validation of the work is presented via a simplified version of SCADA (Supervisory Control and Data Acquisition) system case study.**

***Keywords*-Real-time, Security patterns, Schedulability analysis, Model-Driven Engineering (MDE).**

I. Introduction

Our society has become more dependent on software-intensive systems, such as Information and Communication Technologies (ICTs) systems, not only in safety-critical areas but also in areas such as finance, medical information management and systems using web applications. The complexity of such systems during their design comes from the involvement of transdisciplinary concerns. Indeed, such systems must satisfy a number of requirements (real-time, physical, energy efficiency and others). In addition, these systems have to satisfy assurance requirements (e.g., IEC 61508 and ISO 27005 [1], for dependability and security concerns). This brings the complexity of such systems to a higher level. In particular, security concerns have an impact on other concerns such as real-time performance. Therefore, architects must apply trade-offs to satisfy functional requirements (real-time), and security requirements as two categories of constraints.

Model-Based System Engineering (MBSE) provides a useful contribution for the design and evaluation of secure systems. It makes easier the enactment of the separation of concern paradigm (security, real-time, performance, etc.). It helps the architect specify in a separate view non-functional requirements such as security at a high level of abstraction. Moreover, expertise and knowledge in system architecture and security can be captured within patterns that provide generic solutions for recurring problems. In particular for security, where protecting data and services is an important issue, security pattern catalogs [2] provide guidelines to build secure architectures.

This work is part of a more general process devoted to incremental pattern-based modeling and safety and security analysis for correct by construction systems design. In previous works, we have proposed a model-based approach for guiding the selection of security patterns based on risk analysis and pattern classification [3]. More recently in [4], we proposed an approach to support Security, Dependability and Resource Trade-offs using Pattern-based Development and Model-driven Engineering. In this paper, we go one step further, we study the impact of implementation alternatives of these security solutions onto the system architecture. A special emphasis is paid to timing performance concerns using model-based real-time evaluations. In this context, the system architect starts from a functional architecture and an abstract platform. The artifacts are abstract at this stage of development but contain temporal information (e.g., computation cost, deadlines and period of event for each function). Once the security requirements are specified, several security pattern solutions are proposed from a repository of patterns. The real-time evaluation helps the architect to select the best candidates that respect timing concerns (e.g., maximum utilization capacity in the platform). An evaluation of the proposed approach is presented through its practical application on a SCADA system case study, which has strong security requirements, to support a pattern-based development approach.

The remaining sections are organized as follows: Section II presents a global picture of the positioning of this work. Section III presents the main steps of the real-time evaluation approach of security solution alternatives. Section IV presents the model-based framework with the used modeling languages, the model transformations and the tool support. Section V discusses the obtained experimental results applied to a SCADA (Supervisory Control and Data Acquisition) system case study. Section VI positions the paper towards related works and section VII concludes the paper and discusses future work.

II. Background

The general approach consists of building secure software and systems at high level design stage using patterns as its primary technique: Patten Based System Engineering (PBSE) [5], [6]. Fig. 1 shows an excerpt of the approach which consists of three collaborative processes: architecture design, risk management and pattern solution. At the architecture design process, first a conceptual model of the system

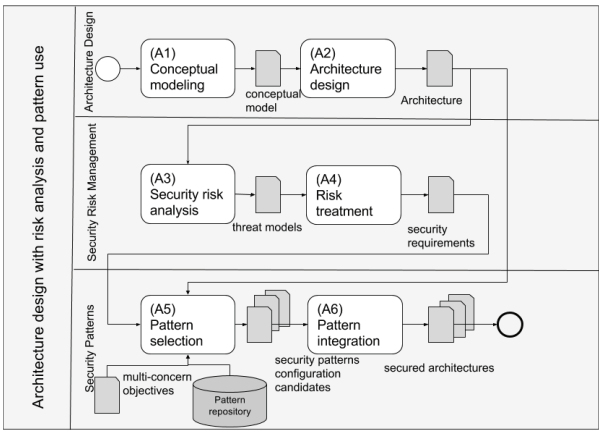

Figure 1. Architecture design with risk analysis and pattern use

is designed (A1) then the system architecture is designed describing the functions of the system (A2). In parallel, in the risk management process, the system architecture is submitted to risk analysis in order to enumerate threats (A3) which are treated to derive security requirements (A4). This process is done by a security risk analyst. Security requirements describe what should be provided by security mechanisms in order to stop some threats. They do not deal with how they are implemented. The search of solution patterns relies on SEMCO [7] which provides two levels of descriptions: abstract and concrete. At the pattern solution process, abstract patterns are selected according to security requirements. When dealing with implementation, concrete patterns i.e., refinements of abstract patterns, are selected according to hardware/software resource constraints (A5). Pattern integration (A6) consists on applying the identified patterns on the system architecture. There may be several sets of security patterns. In this paper we focus on activity (A5).

III. Schedulability analysis of security pattern configurations

A. Methodology description

In this section we present an overview of the proposed methodology in Fig. 2. The main objective of the workflow is to support real-time evaluation of various possible security pattern configurations that we will call “security solution alternatives" to assess their soundness regarding real-time constraints. This seamless process relies on three main kinds of artifacts: (1) functional architectures to describe system and software functions, and (2) security patterns to describe system security solutions and (3) platform models to describe hardware resources. These concepts have been defined in [4].

As described in Fig. 2, during the first step, the architect first provides the functional architecture specification model of the real-time system together with a security pattern configuration model. The functional specification contains timing parameters: end-to-end flow deadlines, function execution time budgets, activation event patterns (periodic, aperiodic, sporadic). A security patterns configuration is a subset of a system of patterns composed of a list of security patterns and a list of relationships between these security patterns. It will be used to specify one possible structure of an application based on security patterns which will be deployed on a platform [4]. Next, the designer identifies the merge points as prerequisite to establish *bindings* (represented as dashed arrows in Fig. 2). These two models accompanied with the bindings are used as inputs to the “Pattern Configuration Integration” (Step 2). The output is a refined functional architecture specification with added security patterns functions. Step 3 is responsible for the creation of task model based on on the functional architecture, the platform specification and the mappings between them. Using the resulting task model, schedulability analysis with offset-based scheduling [8] is performed (Step 4). If the task model is schedulable then the pattern configuration is added to the set of candidates (Step 5). Otherwise, the architect rejects this configuration and continues evaluating another one. The next sections describe the used algorithms for each step.

B. Pattern configuration integration

Bindings are necessary for linking functions in the functional architecture to those in the pattern. Let Ma be a functional architecture specification, C a pattern configuration and B the bindings between them. To define the refined functional architecture obtained by integration, we define the algorithm in Listing 2 called *IntegratePatternConfiguration*.

```
1  Algorithm IntegratePatternConfiguration
2      Input: Ma, C, B.
3      Output: Mac.
4   for each Pattern in the C  do
5   for each bi in B do
6    if bi.patternFunction.pattern  = Pattern
7     substitute(Mac,bi.applicationFunction, bi.
          patternFunction)
8     save()
9    endif
10  endfor
11  endfor
```

Listing 1. Pattern Configuration integration algorithm

The pattern configuration and the set of bindings are parsed and for each pattern in the configuration, equal bindings (originating from a function owned by the pattern) are looked up (line 4-6). The integration is then proceeded by substituting the targeted function by the binding i.e., application function ($bi.applicationFunction$) with the source of the binding i.e., pattern function ($bi.patternFunction$) (line 7-8). In our case, the integration process is a merge. A more comprehensive integration method has been studied in [9].

C. Task model generation

The generation process starts by eliciting all end-to-end flows in the functional architecture. An end-to-end flow is a set of communicating function from one end to the other. For example, in Fig. 2, there are two possible end-to-end flows: “E1” and “E2”. Each each end-to-end flow in the

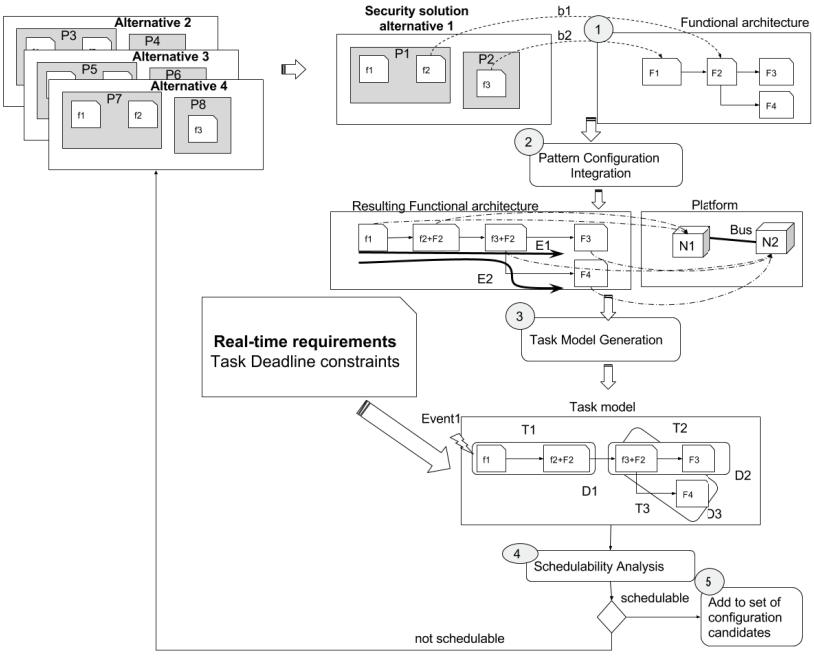

Figure 2. Schedulability analysis of security pattern configuration

functional architecture is a potential task if and only if all the functions are allocated in the same platform node. Otherwise, each set of communicating functions in the end-to-end flow allocated in the same node are assigned to one task. There is a chance that functions can shared between several tasks, in this case the function can only be called by one task and this adds a blockage time. In Fig. 2, in end-to-end flow "E1", functions ("f1","f1+F2")/("F3","f3+F2") are allocated in nodes "N1"/"N2" respectively. Thus two set of functions are assigned to two different tasks "T1" and "T2". Note, however, it is possible to integrate other task model techniques such as those dealing with optimization [10].

D. Real-time evaluation of pattern configurations

Let M be a set of type Mac (i.e., the set of functional architecture specification obtained by integration a pattern configuration) and T of type $TMac$ (i.e., the set of task models for each architecture in Mac. To define the set of schedulable architecture configurations M, we define the algorithm defined in Listing 2 called *EvaluateArchitectureConfiguration*.

```
1  Algorithm EvaluateArchitectureConfiguration
2      Input: T.
3      Output: M.
4   for each TMac in T  do
5   schedulabilityAnalysis(T);
6   if T.isSchedulable
7    M:= M U TMac.Mac
8    save()
9   endif
10 endfor
```

Listing 2. Real-time evaluation of pattern configurations algorithm

T is parsed and each task model $TMac$ is submitted to schedulability analysis. If the $TMac$ passes the test then it is added to M. Otherwise it is rejected.

IV. Model-based development

In this section we describe an MDE framework to support the previous approach. We use metamodeling, existing modeling languages and model transformation techniques for the specification and analysis of secure system and software architecture. However, the approach does not prescribe a fixed set of metamodels and model transformations to be used. As mentioned earlier, here we only focuses on security requirements that directly influence timing constraints.

A. A Metamodel for S&D patterns (SEPM)

The System and Software Engineering Pattern Metamodel (SEPM) [11] is a metamodel for describing Security and Dependability (S&D) patterns, and constitutes the base of our pattern modeling language. Here we consider patterns as sub-systems that expose services (via interfaces) and manage S&D and Resource properties (via features) yielding a unified way to capture meta-information related to a pattern and its context of use. The following paragraph details the principal concepts of the SEPM metamodel to specify an S&D pattern, as described with Ecore notations in Fig. 3.

- *SepmPattern.* This block represents a security pattern as a subsystem describing a solution for a security particular recurring design problem that arises in specific design context.

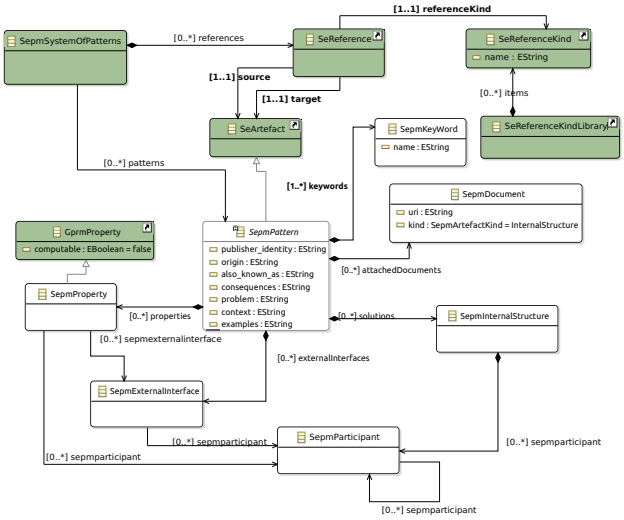

Figure 3. The (simplified) SEPM Metamodel

- *SepmExternalInterface.* A *SepmPattern* interacts with its environment with *Interfaces*.
- *SepmProperty.* A property denotes a particular characteristic of a pattern related to the concern it is dealing with and dedicated to capture its intent in a certain way. For instance, security and dependability properties (SnDProperty)and resource properties (ResourceProperty).
- *SepmInternStructure.* This constitutes the implementation of the solution proposed by the pattern. How the participants collaborate to carry out their responsibilities for the realization of the solution.
- *SepmParticipant.* A listing of the component used in the pattern and their responsabilities in the design. In our context, a participant is a component type with a security-specific purpose. It's role is to add new functionality to the system that is specific to a security requirement the system should uphold. In the context of this study, we use the term *Function* to refer to this concept.
- *SepmSystemOfPatterns.* A pattern system is a set of individual pattern with their relationships (References). Thus dependencies between specific problems can be considered in a comprehensive way.
- *SeReference.* This link is used to specify the relationship between patterns with regard to the domain and software lifecycle stage in the form of a pattern language. For example, a pattern at a certain software lifecycle stage uses another pattern at the same or at a different software lifecycle stage. *SeReferenceKind* contains examples of these links. Here, we create the *SeReferenceKind* model library to support the specification of relationships across artifacts (e.g., refines, specializes and uses) as an extension of the relationship classification proposed in [12].
  - *refines.* It is used to represent the refinement relationship between two patterns.
  - *specializes.* It is used to represent the specialization relationship (detail).
  - *uses.* It is used to represent the functional dependency relationship between two patterns.
  - *isSimilar.* It allows to link two patterns that perform the same functionality. This link is often used to link software patterns to their equivalent hardware patterns.
  - *isAnAlternative.* It allows to link tow patterns that solve the same problem, but propose different solutions.

Example. We illustrate the usage of the SEPM for specifying a pattern with the example of secure communication pattern based on SSL[1] mechanism. Here, we specify an S&D property: "authenticity of sender and receiver". To type the category of this property we use a category from the earlier defined in the S&D category library: *Authenticity*. Moreover, we identify some resource properties, such as "CPU resource time for encryption" and "CPU resource time for authentication" that belong to category *CPUTime*, and "extra energy cost for encryption" and "extra energy cost for authentication" that belong to category *PowerConsumption*.

B. UML + MARTE

The functional architecture and platform are modeled using UML language [13]. In addition, we use a subset of MARTE profile to annotate the models. MARTE standard [14] provides concepts for the modeling and analysis of real-time embedded systems (RTES) and CPSs. Table I shows the used concepts.

MARTE Stereotype	UML extension
Functional architecture stereotypes	
GQAM::GaWorkloadBehavior	Activity
GQAM::GaWorkloadEvent	AcceptEventAction
SAM:: SaEndToEndFlow	ActivityPartition
SAM::SaStep	CallActionBehavior
Alloc::Allocate	Abstraction
Alloc::Allocated	CallAction, Property
Platform Stereotypes	
GQAM::GaPlatformResources	Class
SAM::SaExecHost	Property
SAM ::SaCommHost	Connector
SAM::SaSharedResource	Property
GRM::SchedulableResource	Property

Table I
MARTE ANNOTATIONS

C. M2M transformation: SEPM to UML+MARTE

1) Pattern instantiation: Patterns need to be instantiated from the model-based repository SEMCO described in SEPM into the UML-based development environment. For this purpose, Table II shows transformation rules using the source Metamodel (SEPM) and the target Metamodel (UML) with MARTE annotations.

[1]The TLS Protocol Version 1.2. rfc5246, 2008.

2) Pattern integration: Once the instantiation of a security pattern configuration is done, patterns need to integrated into the functional architecture. This is done with model merge techniques. Now that all the models are in the UML-based modeling environment, the functional architecture together with the security pattern models are composed of a set of UML::Property annotated with SAM::SaStep representing functions. These elements contain the value of their execution time. Once the bindings between the architecture and the security pattern configuration are done; the transformation rules add to the linked architecture functions the value of the execution time of the related security pattern functions. It also adds new functions introduced with the patterns. A new architecture configuration is obtained.

Source	Target		
SEPM	*UML*	*MARTE annotations and types*	*Comments*
SepmPattern	Package	N/A	This package is the main container
Context	Comment	N/A	The textual description is put in this comment inside the UML package
Problem	Comment	N/A	—
SepmInternalStructure	Class	N/A	The class describes the functional architecture of the pattern and contains functions
Function	Property	SAM::SaStep	The functions are put inside the container class. Each SEPM::Function has a S&D property and/or a resource property
ResourceProperty	Class	NFP_Real	This mapping depends on the resource property category and thus the target is one of the sub types of NFP_Real. E.g., "CPUTime" property is mapped to NFP_Duration (sub-type of NFP_real).

Table II
M2M TRANSFORMATION RULES FROM SEPM TO UML+MARTE

D. Modeling process

The modeling process follows the steps described in [15] based on MARTE. The goal is to build a task model from the design in order to evaluate and compare architectural solutions. We describe here the main modeling steps and position them with regards to steps 2, 3 and 4 in Fig. 2.

The input is a functional view of the application obtained by integration a security pattern configuration using UML and MARTE annotations (result of step 2 in Fig. 2). A task model

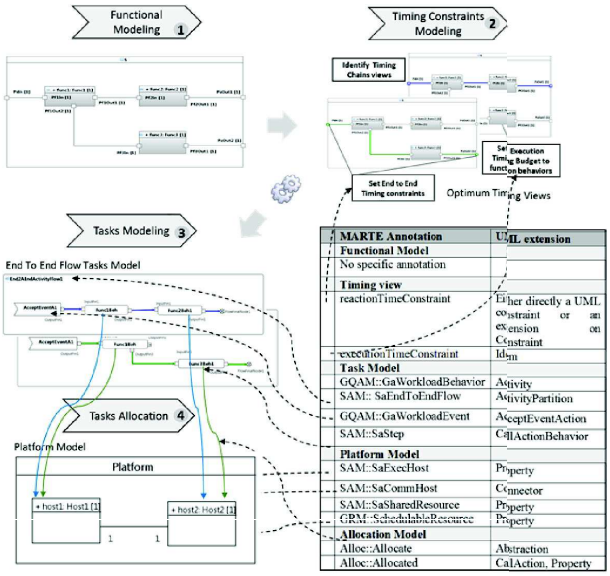

Figure 4. Using MARTE to set timing constraints

is obtained following four steps: (1) identification of event-chains in the functional model and (2) specification of timing constraints (step 3 in Fig. 2), and (3) computation of a MARTE task model and (4) task model allocation (step 4 in Fig. 2). The workflow and MARTE notations used are summarized in Fig. 4.

- *Event chains identification.* The functional organization of the application is described in a Composite diagram showing functions and their connections. From this global view several timing views corresponding to end-to-end flows are selected.
- *Timing constraints setup.* Selected event chains are then tagged to setup timing constraints. MARTE annotations are added to these diagrams to set: (1) event chains timing constraints (between 2 ports), (2) execution time constraints on functions (actually expected for the behavior implementing the function).
- *Task model setup.* The task model structure is described using activity diagrams and can be directly obtained from the event chains specifications above. Each of them is translated into a MARTE end-to-end event flow. Each flow is activated by the reception of an event and described by the consequent behaviors implementing the various functions traversal connected through connectors. MARTE annotations are used to: (1) characterize a timing configuration, (2) specify a data arrival pattern for the activating event (workflowEvent) and (3) specify constraints on the different steps (behaviors involved in the event flow).

- *Allocation model setup.* Finally an allocation model is described in a composite diagram that shows the allocation between functions (actually the tasks corresponding to their behaviors) onto a platform model. MARTE annotations are used to: (1) set allocation relations and (2) set hardware architecture characteristics on execution hosts and communication channels.

E. Design and analysis tools

As discussed below, the proposed tool chain is designed to support the proposed metamodels and model-transformations. Appropriate tools for supporting the approach must fulfill the following key requirements:

- Enable the creation of the UML models used to describe system and software architecture.
- Allow the creation of a custom UML profile.
- Support the implementation of a repository to store pattern models and the related model libraries for classification and relationships.
- Enable the creation of visualizations of the repository to facilitate its access.
- Support the access to the repository. Create views on the repository according to its APIs, its organization and the needs of the targeted system engineering process. For instance, a keyword-based search access tool is provide within the semcomdt suite.
- Enable transformations of the pattern models from the repository format into the target-modeling environment (Papyrus pattern format).
- Enable the creation of system of pattern models in the target-modeling environment.
- Enable the creation of pattern configuration models in the target-modeling environment.
- Enable the integration of application models and models imported from the repository.
- Support the calculation of resource consumption and real-time scheduling.
- Provide the ability to create customized reports by querying the resulting models.

Amongst the existing alternatives, we have chosen Papyrus UML [2]. Specifically, we used Papyrus to create the UML diagrams for the model of the application. SEMCOMDT [3](SEMCO Model Development Tools, IRIT's editor and platform plugins) is used to support pattern repository. The generation of pattern system configuration from a pattern system has been described in [4]. The MARTE UML profile is already integrated into Papyrus. Only stereotypes described in section IV-B are used. The model transformation to support: (1) the instantiation of the patterns form the model-based repository SEMCO into the modeling environment; and (2) the integration of the patterns in the architecture are implemented using QVT Operational language [4]. Real-time analysis have been performed using under development tool called "Qompass Architect" [16]. It is a model-based tool developed in CEA LIST for QoS assessment and optimization of real-time architectures. Qompass Architect explores non-functional properties of real-time architectures to finally synthesize an optimized architecture. Note that other tools performing schedulability analysis can be used such as cheddar[17].

[2]https://eclipse.org/papyrus/

[3]http://www.semcomdt.org

[4]https://projects.eclipse.org/projects/modeling.mmt.qvt-oml

V. Case study

As a preliminary experiment, we apply the approach to a SCADA system case study.

A. Description

SCADA systems are meant to control processes through local controllers, acquiring field data and returning them to a SCADA master computer system. Fig. 5 shows a typical SCADA system architecture. It consists of a SCADA master, an operator workstation and a number of field devices connected by a communication infra-structure. Field devices can be Programmable Logic Controllers (PLC), Remote Terminal units (RTU), sensors and actuators.

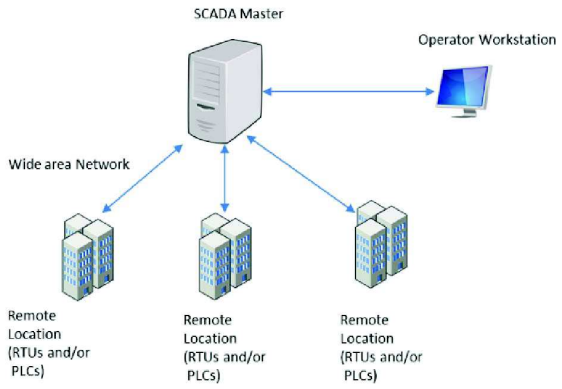

Figure 5. A typical SCADA system architecture [18]

The SCADA master provides the operator with a Human-Machine Interface (HMI) through a work station to issue commands to PLCs and gather field data from them. PLCs are digital computers programmed to continuously monitor sensors and control actuators (e.g., valves, pumps, etc.). RTUs are used for converting sensor data into digital data. As SCADA systems cover large areas, they use Wide Area Networks (WAN). SCADA systems provide the following features: data acquisition and handling (e.g., polling data from controllers, alarm handling, calculations, logging and archiving) on a set of parameters, typically those they are connected to.

B. SCADA systems security

Several techniques can be used we can to enumerate threats and thus derive the security requirements: attack trees, misuse cases and misuse activities. Table III shows a list of some of the risks targeting the assets for such SCADA systems [2].

For this case study, only the following requirements are considered:

Assets	Threats
SCADA master	Physical attacks Malicious settings of the field units Wrong commands sent to the field units Malicious alteration of the parameters of the SCADA master Denial of service
Communication	Sniffing commands Spoofing Denial of service
PLC	Physical attacks Malicious alteration of the run-time parameters Incorrect commands sent to the central controller Malicious alarms sent to the central controller Denial of service

Table III
SCADA SYSTEMS THREATS

- *Req.1*. There should be mechanism for secure communication that guarantees data integrity, confidentiality and authenticity.
- *Req.2*. There should be a mechanism that protects against denial of service attacks at the level of the SCADA master.

C. SCADA system architecture

Fig. 6 shows the input functional architecture together with hardware platform. The functional model contains ten functions in three transactions with their deadlines and trigger periods. The hardware topology in the platform contains a SCADA master and a PLC connected with Modbus. The partitioning of functions into tasks and assignment of tasks onto hosts is also showed. In addition the signal between "Set point processing" and "Command computation" is mapped onto a message. The execution budgets of the functions, the assigned tasks and hosts are showed in Table IV. The values of the SCADA function timing parameters are based on IEEE 1646 standard [19] specifying communication deadlines and IEC 61850 [20] specifying communication network delays in different information categories.

D. Identification of security pattern alternatives

After analyzing security requirements, the architect identifies a set of security patterns along with their refinement alternatives, i.e., concrete patterns. It is important to note that the selection of security patterns takes into account conflicts due to inconsistencies between patterns. For example, Limited view and Full view pattern are conflictual by nature so that implementing both of them in a system will surely bring inconsistencies. The search in the model-based repository leads to the identification of two abstract patterns refined by concrete ones:

- *SecureComm pattern*[2]: ensures that data passing across a secure network is secure. It can be refined by two patterns: SecureCommSSL (P1) and Secure-CommIPsec (P2). SecureCommSSL uses X.509 certificates for authentication and secure channel for creating a cryptographic tunnel.
- *Firewall pattern*[2]: restricts access to internal networks which can be refined by PacketFilter (P3) and Stateful-Filtering (P4).

The result of this step is the System of security patterns represented in Fig. 7.

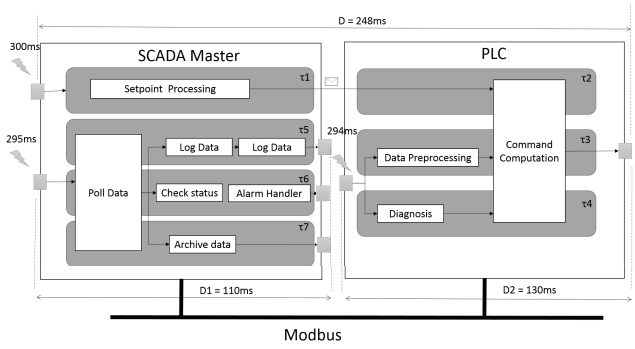

Figure 6. Input functional architecture, hardware platform and deployment

Functions	Execution time	Task	Host
Setpoint Processing	8.7	$\tau 1$	SCADA master
Poll Data	9.6	$\tau 1$, $\tau 6$, $\tau 7$	SCADA master
Log Data	8.5	$\tau 5$	SCADA master
Check Status	9.6	$\tau 6$	SCADA master
Visualize Data	10.5	$\tau 5$	SCADA master
Alarm Handler	10.3	$\tau 6$	SCADA master
Archive Data	9.5	$\tau 7$	SCADA master
Command Computation	10	$\tau 2$, $\tau 3$, $\tau 4$	PLC
Data Preprocessing	9.5	$\tau 3$	PLC
Diagnosis	8.9	$\tau 4$	PLC

Table IV
TIMING PARAMETERS AND DEPLOYMENT OF SCADA FUNCTIONS

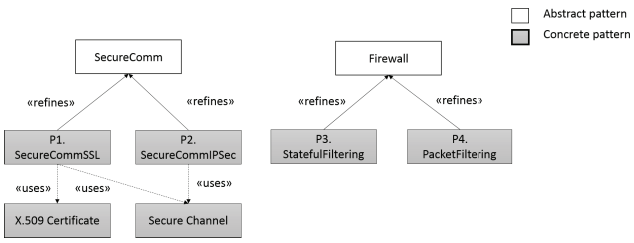

Figure 7. System of security patterns

Fig. 8. shows the corresponding possible security solution alternatives. Each system of security solution alternative consists of a set of concrete patterns in dark grey.

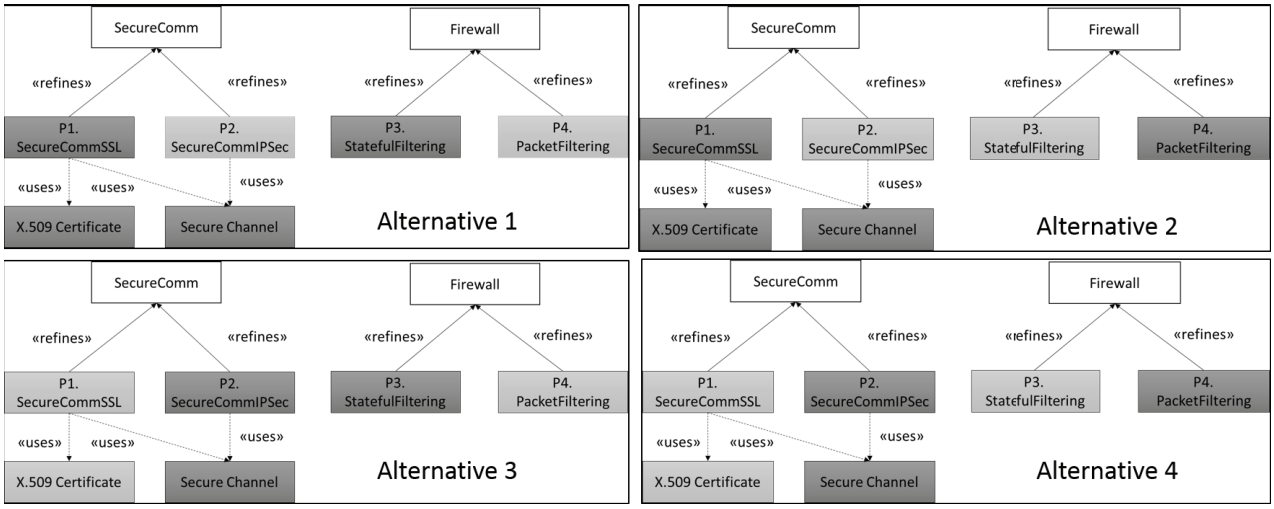

Figure 8. Security solution alternatives generated from the pattern system

The timing and placement parameters of the used security patterns are showed in Table V. The concrete patterns have the same functions but have different execution times. The timing parameters are based on a review of technical reports of SSL/IPsec [21], and stateful/packet firewall [22]. One important point is that the experiment has required some effort in quantifying real-time parameters of security pattern functions. Some functions execution times were estimations and averages. For example, in SecureComm pattern function "HMAC" does not have the same execution time as it depends on the used algorithm (e.g., HMAC-SHA-1-96, HMAC-MD5). However, we believe that estimations and averaging is enough as the approach is meant for high level evaluation and architecture decision making. For example, if none of the security solution alternatives respected the timing requirements because of overload; the architecture of SCADA can be rethought leading to adding an execution node.

Patterns	**Functions**	**Execution time**		**Task**
		(1)	(2)	
SecureCommSSL (1) SecureCommIPsec (2)	Authentication	9.7	38.7	$\tau 1$
	Key exchange	10.1	39.6	$\tau 1$
	Encryption	9.9	9.9	$\tau 1$
	HMAC	9.2	9.2	$\tau 1$
	Decryption	10.3	10.3	$\tau 2$
	Integrity checking	10.2	10.2	$\tau 2$
PacketFiltering (1) StatefulFiltering (2)	Filtering	10	40	$\tau 7$

Table V
TIMING PARAMETERS AND DEPLOYMENT OF SECURITY PATTERN FUNCTIONS

E. Schedulability analysis of security pattern alternatives

The preliminary analysis consists in evaluating the placement of SCADA and pattern functions on hosts described in Table IV and Table V for each security solution alternative (1, 2, 3 and 4) in Fig. 8.

The left side of Fig. 9 shows the node utilization results of each security solution alternative. The utilization bound of the SCADA master and PLC are up to 75.68% (four tasks) and 77.97% (three tasks) respectively. Security solution alternatives 2 and 4 are rejected because the SCADA master utilization in the two cases (83.33% and 103.33%) exceeds the threshold. Response time analysis given in [8] is performed from security solution alternatives 1 and 3 since they pass the preliminary evaluation. Task response time is up to 280ms in security solution alternative 3 and violates its deadline of 248ms. This is due to the offset added by task and the message transmission time. All tasks of security solution alternative 1 respect their deadline: (150ms), (240ms), (60ms), (120ms), (70ms), (100ms) and (30ms). From the evaluations, security solution alternative 1 fulfils security requirements and respects real-time constraints.

F. Discussion

From this first experimentation, we conclude that the approach fulfils the objective of finding a set of security patterns respecting real-time constraints. The work has two main contributions: (1) the proposal of abstract security pattern solutions fulfilling security requirements and (2) the evaluation of the possible implementations fulfilling real-time requirements by the integration of possible security solution alternatives. In this context, this work can be beneficial to resource constrained embedded systems e.g., automotive, avionics. For instance in EAST-ADL [23], trade-off analysis is performed for one design model with different parameters whose values determine whether the design satisfies the model or not. Our work adds a

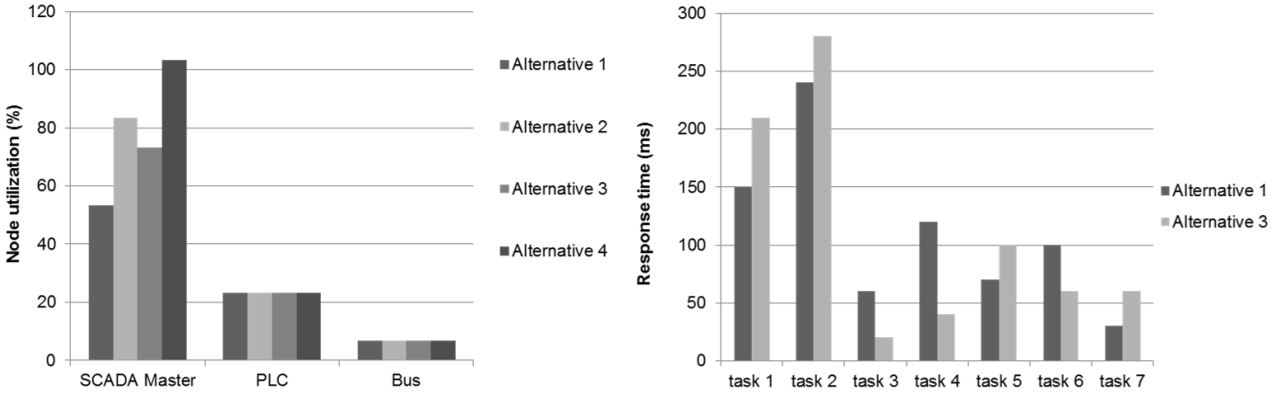

Figure 9. Node utilization for security solution alternatives 1, 2, 3 and 4, and tasks response times for alternatives 1 and 3

step forward which is the evaluation of different design alternative models against non-functional concerns (security in this paper). This work can benefit from EAST-ADL in managing security solution alternatives by using features diagrams. The work can also use task optimization techniques such as the one defined in [10].

VI. Related Work

The evaluation of security solutions with regards to quality attributes is not new. However the evaluation of the integrated security solution with regards to the overall system architecture quality and performance is a fresh topic. This paper contribution is a step towards this goal. It presents a model-based method for evaluating security pattern-based solutions for decision purposes. The evaluation was done in the context of temporal evaluations. The evaluation results at each step guide the architect to select the best possible alternative.

A. Security and Real-Time Requirements

Previous work have focused on the security and real-time requirements separately: dependability and security modeling and analysis[24][25] and real time requirements [26][27]. A survey of dependability modeling and analysis frameworks with UML can be found in [24]. It focuses on software systems Reliability, Availability, Maintenance and Safety (RAMS). In [25], the authors have extended MARTE with a Dependability Analysis and Modeling (DAM) UML profile and applied it to an intrusion-tolerant message service case study. In [26], the authors presented a staged approach to optimize the deployment in the context of real-time distributed systems.

B. Architecture Optimization, Decision and Trade-off Analysis

Other works focused on large scale architecture optimization, decision and trade-off analysis [28][29][30][31]. In the automotive domain, a multi-objective automatic optimization approach based on EAST-ADL modeling is proposed [28]. It supports the evaluation of alternative architectures according to dependability, timing performance, cost etc. A similar work in [29] presented a method for the search of optimal architecture design according to multi-objectives such as cost, performance and reliability based on SysML modeling. In [30], the authors identified limitations in UML language to support architecture decision according to non-functional attributes. They propose a framework based on parametric analysis specification that aims at evaluating design decisions. More specifically in security and performance interplay, the study in [31] focused on the analysis of the performance effects of security solutions modeled as UML non-functional aspects. It used SPT UML profile for annotating a UML design with schedulability, time and performance data. The resulting model and the security aspects were transformed separately and composed into one model which is then analyzed.

VII. Conclusion

The paper presents a model-based approach for evaluating security solution alternatives against real-time requirements. The approach is applied to a SCADA system case study which shows the applicability of the approach. The main benefits are to provide a tooling support to allow early evaluation of different implementation of security measures. This work is part of a process devoted to incremental pattern-based modeling and a safety and security analysis for correct by construction systems design. The results obtained help the designer select appropriate design solution to reinforce security. The methodology relies on UML/MARTE for modeling and makes extensive use of MARTE to perform architectural evaluation for timing concerns, this will be extended in the future to address other concerns (e.g., cost, reliability, memory consumption, power supply). The next step is to do such evaluation and use trade-off analyses to decide between alternative pattern solutions.

References

[1] Information technology – Security techniques – Information security risk management, http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=56742, [Accessed: April-2016] (2011).

[2] E. B. Fernandez, Security patterns in practice: Building secure architectures using software patterns, Software Design Patterns, Wiley, ISBN 978-1-119-99894-5, 2013.

[3] A. Motii, B. Hamid, A. Lanusse, J.-M. Bruel, Guiding the Selection of Security Patterns Based on Security Requirements and Pattern Classification, in: Proceedings of the 20th European Conference on Pattern Languages of Programs, EuroPLoP '15, ACM, 2015, pp. 10:1–10:17.

[4] B. Hamid, Interplay of Security&Dependability and Resource Using Model-Driven and Pattern-Based Development, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, IEEE, 2015, pp. 254–262.

[5] E. B. Fernandez, Using security patterns to develop secure systems, in: Software engineering for secure systems. Industrial and research perspectives, 2011, pp. 16–31.

[6] B. Hamid, J. Geisel, A. Ziani, J.-M. Bruel, J. Perez, Model-Driven Engineering for Trusted Embedded Systems Based on Security and Dependability Patterns, in: SDL 2013: Model-Driven Dependability Engineering, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 72–90.

[7] B. Hamid, Modeling of Secure and Dependable Applications Based on a Repository of Patterns: The SEMCO Approach, Reliability Digest, IEEE Reliability Society, Special Issue on Trustworthy Computing and Cybersecurity 1 (1) (2014) 9–17.

[8] K. Tindell, J. Clark, Holistic Schedulability Analysis for Distributed Hard Real-time Systems, Microprocess. Microprogram. 40 (2-3) (1994) 117–134.

[9] B. Hamid, C. Percebois, D. Gouteux, A Methodology for Integration of Patterns with Validation Purpose, in: Proceedings of the 17th European Conference on Pattern Languages of Programs, EuroPLoP '12, ACM, 2012, pp. 8:1–8:14.

[10] R. Bouaziz, L. Lemarchand, F. Singhoff, B. Zalila, M. Jmaiel, Architecture Exploration of Real-Time Systems Based on Multi-objective Optimization, in: 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS), 2015, pp. 1–10.

[11] B. Hamid, S. Gürgens, A. Fuchs, Security patterns modeling and formalization for pattern-based development of secure software systems, Innovations in Systems and Software Engineering, Springer 12 (2) (2016) 109–140.

[12] J. Noble, Classifying Relationships Between Object-Oriented Design Patterns, in: Proceedings of the Australian Software Engineering Conference, ASWEC '98, IEEE Computer Society, 1998, pp. 98–.

[13] UML 2.5, http://www.omg.org/spec/UML/2.5/, [Accessed: April-2016] (2015).

[14] Uml profile for modeling and analysis of real-time and embedded systems (marte), version 1.1, http://www.omg.org/spec/MARTE/1.1/, [Accessed: January-2013] (2011).

[15] C. Mraidha, S. Tucci-Piergiovanni, S. Gérard, Optimum: a MARTE-based methodology for schedulability analysis at early design stages, SIGSOFT Softw. Eng. Notes 36 (1) (2011) 1–8.

[16] R. T. Kolagari, D. Chen, A. Lanusse, R. Librino, H. Lönn, N. Mahmud, C. Mraidha, M. Reiser, S. Torchiaro, S. T. Piergiovanni, T. Wägemann, N. Yakymets, Model-based analysis and engineering of automotive architectures with EAST-ADL: revisited, IJCSSA 3 (2) (2015) 25–70.

[17] F. Singhoff, J. Legrand, L. Nana, L. Marcé, Cheddar: A Flexible Real Time Scheduling Framework, in: Proceedings of the 2004 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-time & Distributed Systems Using Ada and Related Technologies, SIGAda '04, ACM, 2004, pp. 1–8.

[18] Technical Information Bulletin 04-1, Supervisory Control and Data Acquisition (SCADA) System, https://scadahacker.com/library/Documents/ICS_Basics/SCADA%20Basics%20-%20NCS%20TIB%2004-1.pdf, [Accessed: April-2016] (2004).

[19] IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation, IEEE Std 1646-2004 (2005) 0_1–24doi:10.1109/IEEESTD.2005.95748.

[20] P. CODE, Communication networks and systems in substations–Part 5: Communication requirements for functions and device models, https://iecwebstore.com/p-preview/info_iec61850-5%7Bed1.0%7Den.pdf, [Accessed: April-2016] (2003).

[21] A. Alshamsi, T. Saito, A technical comparison of IPSec and SSL, in: 19th International Conference on Advanced Information Networking and Applications, Vol. 2 of AINA '05, 2005, pp. 395–398 vol.2.

[22] D. Hartmeier, Design and Performance of the OpenBSD Stateful Packet Filter (Pf), in: Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, USENIX Association, 2002, pp. 171–180.

[23] V. Debruyne, F. Simonot-Lion, Y. Trinquet, EAST-ADL - An Architecture Description Language, in: Architecture Description Languages, IFIP The International Federation for Information Processing, Springer US, 2005, pp. 181–195.

[24] S. Bernardi, J. Merseguer, D. C. Petriu, Dependability modeling and analysis of software systems specified with UML, ACM Comput. Surv. 45 (1) (2012) 2.

[25] S. Bernardi, J. Merseguer, D. C. Petriu, A dependability profile within MARTE, Software and System Modeling 10 (3) (2011) 313–336.

[26] A. Mehiaoui, E. Wozniak, S. T. Piergiovanni, C. Mraidha, M. D. Natale, H. Zeng, J.-P. Babau, L. Lemarchand, S. Gérard, A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems, in: SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems, LCTES '13, 2013, pp. 121–132.

[27] M. G. Harbour, J. J. Gutiérrez, J. M. Drake, P. L. Martínez, J. C. Palencia, Modeling distributed real-time systems with MAST 2, Journal of Systems Architecture 59 (6) (2013) 331 – 340.

[28] M. Walker, M.-O. Reiser, S. T. Piergiovanni, Y. Papadopoulos, H. Lönn, C. Mraidha, D. Parker, D.-J. Chen, D. Servat, Automatic optimisation of system architectures using EAST-ADL, Journal of Systems and Software 86 (10) (2013) 2467–2487.

[29] P. Leserf, P. d. Saqui-Sannes, J. Hugues, Multi domain optimization with SysML modeling, in: 20th Conference on Emerging Technologies Factory Automation, ETFA '15, IEEE, 2015, pp. 1–8.

[30] H. Espinoza, D. Servat, S. Gérard, Leveraging analysis-aided design decision knowledge in UML-based development of embedded systems, in: Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge, SHARK '08, 2008, pp. 55–62.

[31] M. Woodside, D. C. Petriu, D. B. Petriu, J. Xu, T. Israr, G. Georg, R. France, J. M. Bieman, S. H. Houmb, J. Jürjens, Performance analysis of security aspects by weaving scenarios extracted from UML models, Journal of Systems and Software 82 (1) (2009) 56–74.